UNIVERSITY OF WATERLOO

The CENTRE for EDUCATION in MATHEMATICS and COMPUTING

2021 Beaver Computing Challenge (Grades 5 & 6)

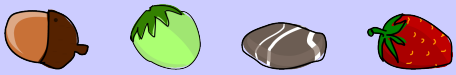
Questions, Answers, Explanations, and Connections

Part A

Strawberry

Story

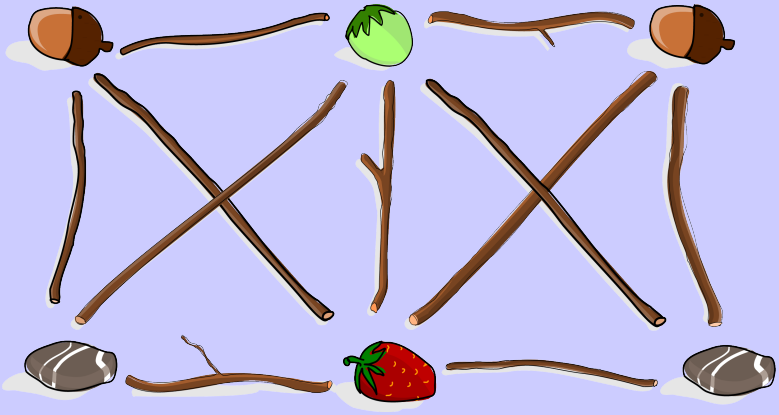
Anja makes a design on the ground using the following four types of objects.



She then places sticks in her design according to her very important rule:

Sticks cannot be placed between objects that are the same type.

Here is Anja's completed design:



Suddenly a bird swoops in and eats the  ! Anja would like to avoid having this happen again.

Question

If possible, Anja would like to replace the  with a different type of object, and without moving any sticks. Without breaking her very important rule, which object can Anja replace the  with?

(A) 

(B) 

(C) 

(D) It is not possible. Only a  could go there.

Answer

(D) It is not possible. Only a  could go there.

Explanation of Answer

Unfortunately, Anja is not going to be able to replace the strawberry with a different type of object, without breaking her very important rule.

In Anja's original design, the strawberry  had sticks between it and every other type of object. Changing the strawberry to anything other than another strawberry would force a stick to exist between two objects of the same type.

Connections to Computer Science

Anja's design is an example of a *graph*. The objects are the *vertices* of the graph and the sticks are the *edges* of the graph. Two objects that have a stick between them, which is equivalent to two vertices that have an edge between them, are called *neighbours*.

The problem that is being asked here is related to *graph colouring*. That is, we try to determine how many colours are needed so that every vertex is coloured a different colour than all of its neighbours. The problem of how to colour a graph using the minimum possible number of colours has many applications, such as scheduling sports competitions, designing a seating plan, and even solving a Sudoku puzzle.

Country of Original Author

Switzerland

Overlapping Coins

Story

Emil has six different coins.

Emil placed the six coins on a table, one at a time. Some coins were placed on top of other coins so that they overlap as shown.

Question

Which coin was the fourth coin that Emil placed on the table?

(A) 

(B) 

(C) 

(D) 

Answer

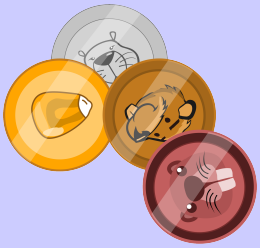
(B) 

Explanation of Answer

To determine the correct answer, we reverse the process.

Notice that the coin  in the bottom-left corner is the only coin that has no other coins on top of it. This means it must have been placed on the table last and was therefore the sixth coin to be placed. Before this coin was placed, the coins on the table must have looked like this:

Now the only coin that has no other coins on top of it is the coin  . This coin must have been placed second to last and was therefore the fifth coin to be placed. Before this coin was placed, the coins on the table must have looked like this:



Now the only coin that has no other coins on top of it is the coin  . This coin must have been placed third to last and was therefore the fourth coin to be placed.

Continuing this process, we find that the coins were placed on the table in the following order:

First Second Third Fourth Fifth Sixth

Connections to Computer Science

The coins in the picture are laid in a *sequence*, and the *order* of the sequence matters.

There are many sequences where the order matters. For example, when getting dressed, putting on shoes should come after putting on pants.

Another example is describing to a computer how to draw a picture. Drawing a circle, then two dots, and then a curved line, will produce a smiley face, as shown in the picture below.

If the order was different, and the circle was drawn last, the two dots and curved line would have been hidden behind the circle.

Computers internally usually also work sequentially. Most computer programs are written so that first one action and then another action happens. So a computer program for drawing a smiley face could look like this:

- draw circle at (0,0) with radius 5
- draw dot at (-2,2)
- draw dot at (2,2)
- draw curved line from (-4,1) to (4,1)

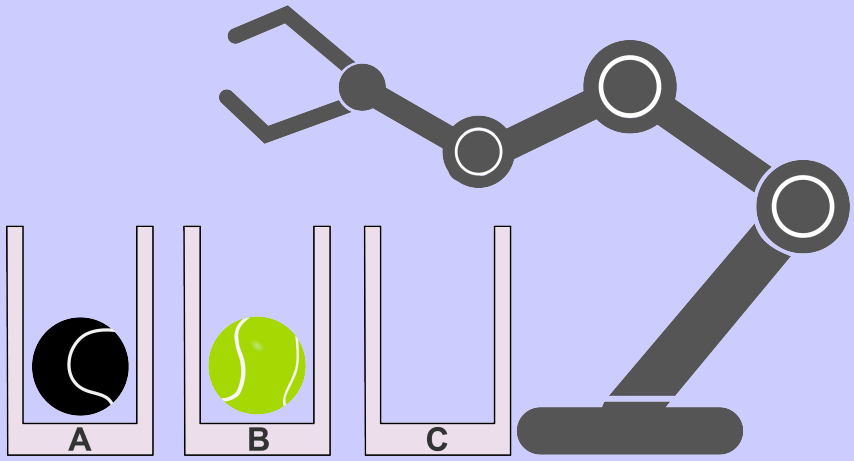
Country of Original Author

Czech Republic

Robot Arm

Story

There are three bins, two balls, and a robot arm that can pick up the balls. Originally, one ball is in bin A and another is in bin B. Bin C is empty.



Then the robot arm completes the following steps in the order given:

1. Pick up the ball in bin A and put it in bin C.
2. Pick up the ball in bin B and put it in bin A.
3. Pick up the ball in bin C and put it in bin B.

Question

When the robot arm is finished, which of the following statements is true?

(A) The ball originally in bin A is now in bin B, and the ball originally in bin B is now in bin A.

(B) Both balls are in bin A.

(C) Bin A is empty.

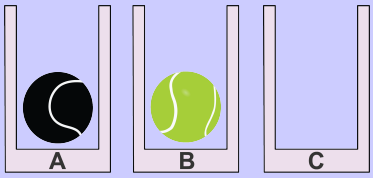
(D) Nothing has changed. Each ball is back in its original bin.

Answer

(A) The ball originally in bin A is now in bin B, and the ball originally in bin B is now in bin A.

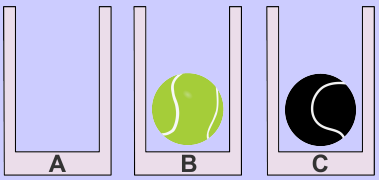
Explanation of Answer

Originally, the balls were in bins A and B as shown.

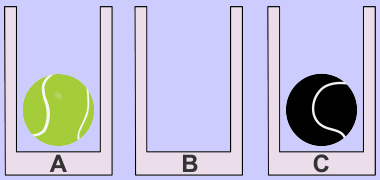


The following diagrams show the contents of the bins after each step.

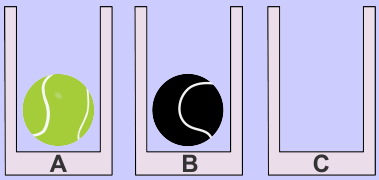
After Step 1:



After Step 2:



After Step 3:



We see that when the robot arm is finished, the ball originally in bin A is now in bin B, and the ball originally in bin B is now in bin A.

Connections to Computer Science

One important aspect of problem solving is keeping track of the *state* of all items that we are interested in. In this problem, the state is the position of the balls in the three bins.

The state of this problem can be thought of as three *variables* each of which represents which ball is in a particular bin. Each movement of the robot arm changes the values of two of the variables: the variable/bin that has a ball removed, and the variable/bin that has a ball added to it.

Keeping track of the state is important in many applications. For example, in solving a sudoku puzzle, playing a game of chess, or the current position on a map app on a phone, keeping track of the state is crucial to determine what the "next" state should be.

When writing a computer program, a programmer has to decide what state(s) they need to keep track of, and use the values of variables to remember what the current state is.

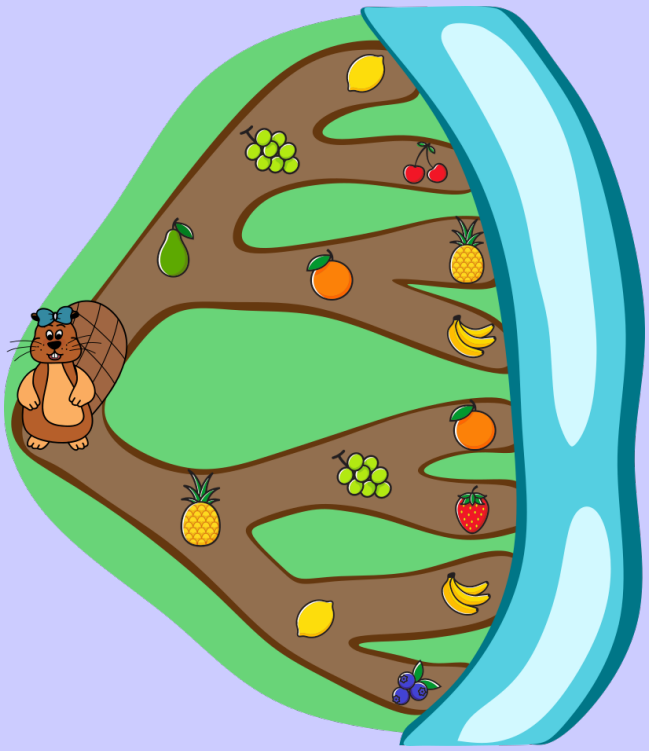
Country of Original Author

Germany

Fruit Road

Story

A beaver wants to take a path to the river. Each path passes by three different types of fruit as shown.



The beaver **must** pass by a pineapple which is its favourite fruit.

The beaver **must not** pass by an orange which it is allergic to.

Question

How many of the eight possible paths can the beaver take?

(A) 2

(B) 3

(C) 4

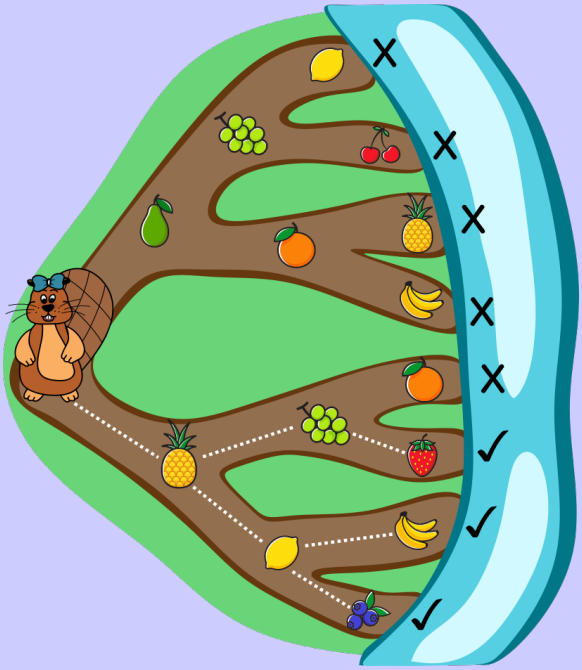
(D) 5

Answer

(B) 3

Explanation of Answer

We consider the paths from the top to the bottom of the diagram. The top two paths do not pass by a pineapple. The next three paths pass by an orange. Only the bottom three paths pass by a pineapple but not an orange.



Connections to Computer Science

For this problem, the beaver is walking through a *directed graph*, specifically, a *tree*.

A *tree* is composed of a *root* node, which is where the beaver begins its journey. Each point where there is a fruit is called a *node* in the tree. The eight fruits near the river are the *leaf* nodes of the tree.

Instead of exploring all eight *paths* in the tree, we can use the technique of *tree pruning* to decrease the number of paths that are explored. Specifically, whenever we encounter an orange on a path, we can avoid searching any of the *subtrees* of that node. This technique is used in *artificial intelligence* to speed up searching through a set of possibilities.

Country of Original Author

Uruguay

Part B

Picking Up Carrots

Story

There are 5 animal homes connected by paths with a carrot on each path, as shown.

Rina Rabbit lives in house R. It takes Rina 1 minute to walk on any path between two homes.

Question

Which of the following routes allows Rina to pick up all the carrots and return home in the shortest amount of time?

(A) $\mathrm{R} \to \mathrm{S} \to \mathrm{T} \to \mathrm{P} \to \mathrm{Q} \to \mathrm{S} \to \mathrm{P} \to \mathrm{R}$

(B) $\mathrm{R} \to \mathrm{P} \to \mathrm{Q} \to \mathrm{S} \to \mathrm{R} \to \mathrm{P} \to \mathrm{T} \to \mathrm{S} \to \mathrm{P} \to \mathrm{R}$

(C) $\mathrm{R} \to \mathrm{S} \to \mathrm{P} \to \mathrm{Q} \to \mathrm{P} \to \mathrm{T} \to \mathrm{S} \to \mathrm{R}$

(D) $\mathrm{R} \to \mathrm{P} \to \mathrm{Q} \to \mathrm{S} \to \mathrm{T} \to \mathrm{P} \to \mathrm{R}$

Answer

(A) R → S → T → P → Q → S → P → R

Explanation of Answer

Since there is a carrot on each path, Rina must walk on each path in order to pick up all the carrots. The route in Option A uses each path exactly once and therefore picks up all the carrots. It is the fastest route because it does not use any path more than once.

For completeness, we will explain why the other routes are not correct.

The route in Option B picks up all the carrots, but it takes more time than the route in Option A because it uses some paths twice.

The routes in Option C and Option D do not pick up all the carrots.

Connections to Computer Science

A graph consists of a set of *vertices* or *nodes* (usually depicted as points or circles) and a set of *edges* (usually depicted as lines, possibly curved, sometimes directed with an arrow). The edges connect vertices. A sequence of connected edges in a graph is called a *path*.

In this problem, the homes are the vertices, and each path with a carrot is an edge. This graph is special, since it is *connected* (there is a path from any vertex to any other vertex by following edges of the graph) and no vertices have an odd number of adjoining edges.

For graphs that have these two special properties, there is a way to follow all the edges in the graph exactly once, which is called an *Eulerian path* named after Leonhard Euler (1707 – 1783) who first described this problem. Euler came up with the concept by trying to solve the problem of the Seven Bridges of Königsberg. Today we can find Eulerian paths by applying the well-known *Fleury's algorithm* or *Hierholzer's algorithm*.

Country of Original Author

China

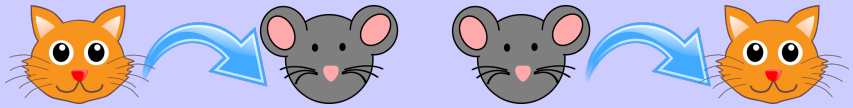
Presents

Story

Friends give presents to one another according to the following two rules.

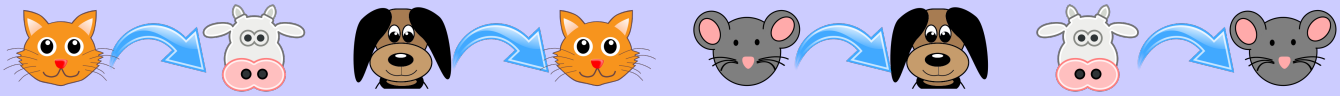
1. Each friend must give exactly one present.
2. Each friend must receive exactly one present.

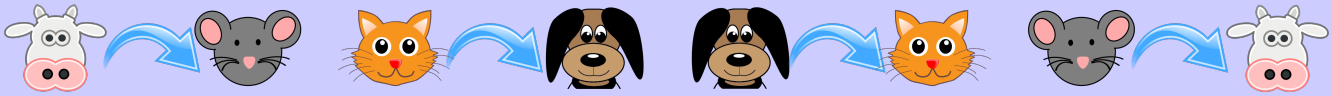
Here is an example where Cat gives a present to Mouse, and Mouse gives a present to Cat:



Question

Cow, Cat, Dog and Mouse give presents to each other. Which of the following options does **not** follow the rules?

(A) 

(B) 

(C) 

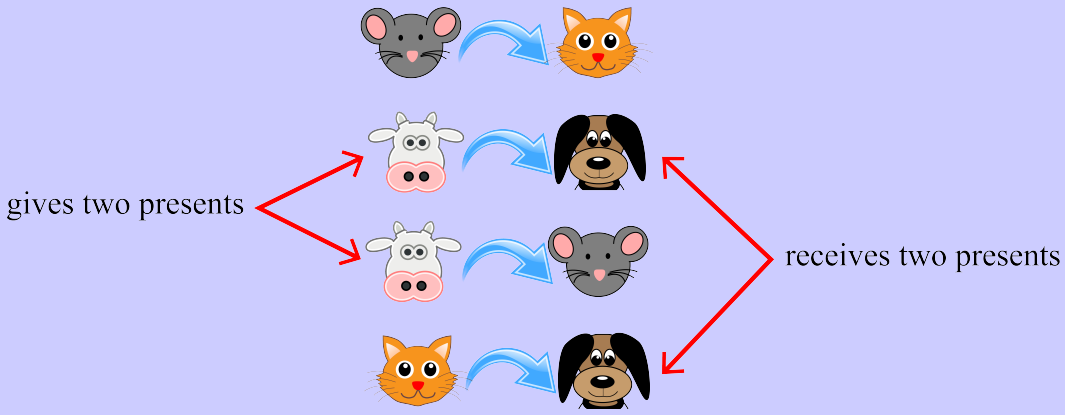
(D) 

Answer

(C) 

Explanation of Answer

Option C does not follow the rules because Cow gives two presents but does not receive any, and Dog receives two presents but does not give any, as shown below.



According to the first rule, each friend must give exactly one present, and according to the second rule, each friend must receive exactly one present, so Option C breaks both rules.

Options A, B, and D all follow the two rules.

In Option A, Cat gives a present to Cow, Cow gives a present to Mouse, Mouse gives a present to Dog, and Dog gives a present to Cat.

In Option B, Cow exchanges presents with Mouse, and Cat exchanges presents with Dog.

In Option D, Cow gives a present to Dog, Dog gives a present to Cat, Cat gives a present to Mouse, and Mouse gives a present to Cow.

Connections to Computer Science

This task can be thought of as a problem of *testing*, and specifically *verification of output*. There are two rules that need to be verified for each sequence: each "giver" must appear exactly once (Rule 1), and each "receiver" must appear exactly once (Rule 2).

Testing is a crucial part of the *software life cycle*. The software life cycle includes

- *analysis*, where the specification is gathered;
- *design*, where the general model and plan for the software is created;
- *implementation*, where the code is written; and
- *testing*, where the code is evaluated and uncovered errors refine the analysis and/or design steps.

The testing process in this problem involves scanning the possible answers to verify that both Rule 1 and Rule 2 are satisfied. In general, the amount of testing increases rapidly as the program code grows and possibly changes over time, due to fixes to software errors or new features being added to the program. Testing can also act as a *certificate of correctness* for both the creators and the users of the software.

Country of Original Author

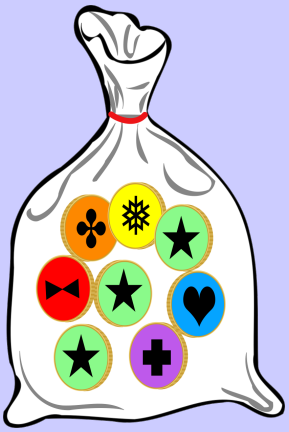
Ireland

Coin Bag

Story

In Saoirse's country there are four different types of coins. Some coins are the same on both sides, and some are not. The images below show both sides of each type of coin.

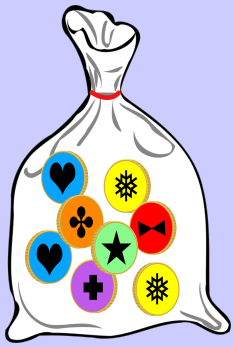
Saoirse has the following bag of coins:

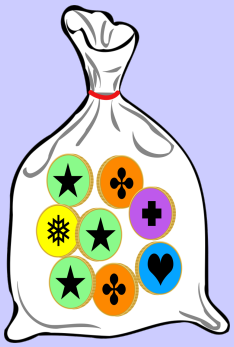


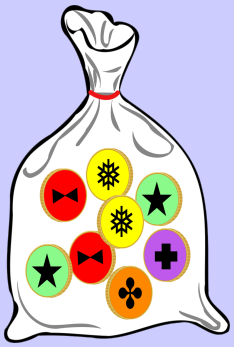
Then the bag is shaken and the coins in the bag move around.

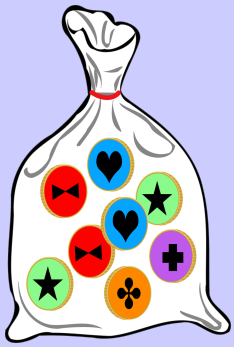
Question

Which of the following could be Saoirse's bag of coins after it was shaken?

(A) 

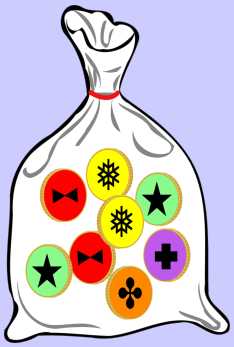
(B) 

(C) 

(D) 

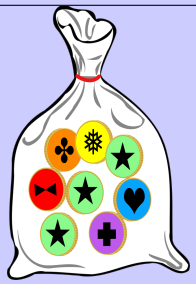
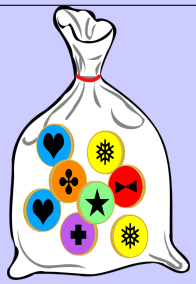
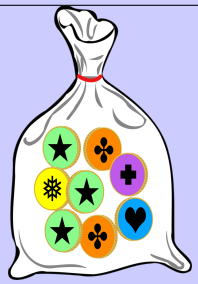
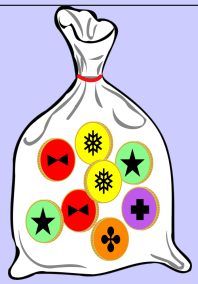
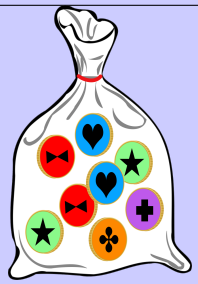
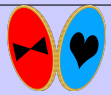
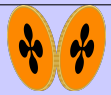
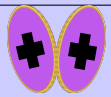
Answer

(C)



Explanation of Answer

Notice that we need to see only one side of each coin to determine which type of coin it is. One way to find the correct answer is to compare the number of coins of each type in Saoirse's bag to the number of coins of the same type in each of the four other bags. This is summarized in the table below.

	 Saorise's Bag	 Bag (A)	 Bag (B)	 Bag (C)	 Bag (D)
	4	3	4	4	2
	2	3	1	2	4
	1	1	2	1	1
	1	1	1	1	1

From the table we can see that Bag (C) is the only bag that has the same number of each type of coin as Saorise's bag. It follows that only Bag (C) could be Saoirse's bag after it was shaken.

Connections to Computer Science

This task is concerned with finding a *bijection* between Saoirse's bag and one of the bags A, B, C, or D. A bijection is a way to pair every element from one set to every element in the other set, so that there are no unpaired elements in either set.

There are $n! = n \times (n-1) \times \cdots \times 2 \times 1$ ("n factorial") different bijections between two sets of size n. In this problem, since there are 8 coins in Saoirse's bag, there are $8! = 40320$ different possible bijections. In order to reduce that number, *constraints* can be applied.

The constraints that can be applied are the number of particular coins of each type in each bag. Specifically, a coin of a certain type must be mapped to exactly the same type of coin in the other bag. This greatly reduces the number of possibilities to examine.

Looking for patterns and eliminating possibilities is a fundamental concept in *constraint programming*, which is a technique used in *artificial intelligence*.

Country of Original Author

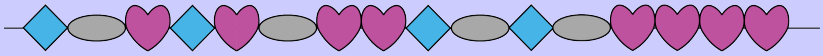
Ireland

Necklaces

Story

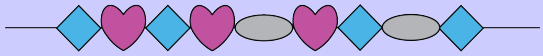
A jeweller makes necklaces with hidden messages by replacing each letter of the alphabet with a bead pattern. Bead patterns are made using heart and diamond beads, and the same bead pattern always represents the same letter. Letters in a message are separated by oval beads and messages are read from left to right.

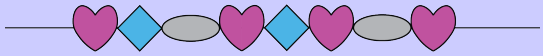
Here are two of the necklaces the jeweller has made along with their hidden messages.

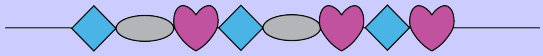
Necklace	Hidden Message
	TRUTH
	CARE

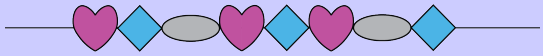
Question

Which of the following necklaces has the hidden message ART?

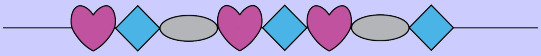
(A) 

(B) 

(C) 

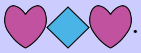
(D) 

Answer

(D) 

Explanation of Answer

The letter A is the second letter in the message CARE, so the bead pattern for A is  .

The letter R is the second letter in the message TRUTH, so the bead pattern for R is  .

The letter T is the first letter in the message TRUTH, so the bead pattern for T is  .

Stringing these together from left to right and separating them with oval beads gives the necklace in Option D.

Though not required, we can use this method to find the hidden messages in the other necklaces as well. Option A contains the message CAT, Option B contains the message ARE, and Option C contains the message TAR.

Connections to Computer Science

This task focusses on *encoding*. Encoding is the process of applying a code to data in order to represent the data in a different but equivalent way. In this task, the encoding of letters is based on the *Morse code*, where the dot(•) of Morse code is replaced by  and the dash(▬) is replaced by .

Morse code is one example of an encoding of letters to a sequence of symbols. Two other common encodings of characters to symbols that are used in transmitting digital messages are *ASCII*, which deals with simple text characters, and *Unicode*, which deals with characters in any language as well as emojis.

How to encode and decode information falls under a field of research called *information theory* which includes the study of *data compression* (how to store data using less memory) and *cryptography* (how to store or send information securely).

Country of Original Author

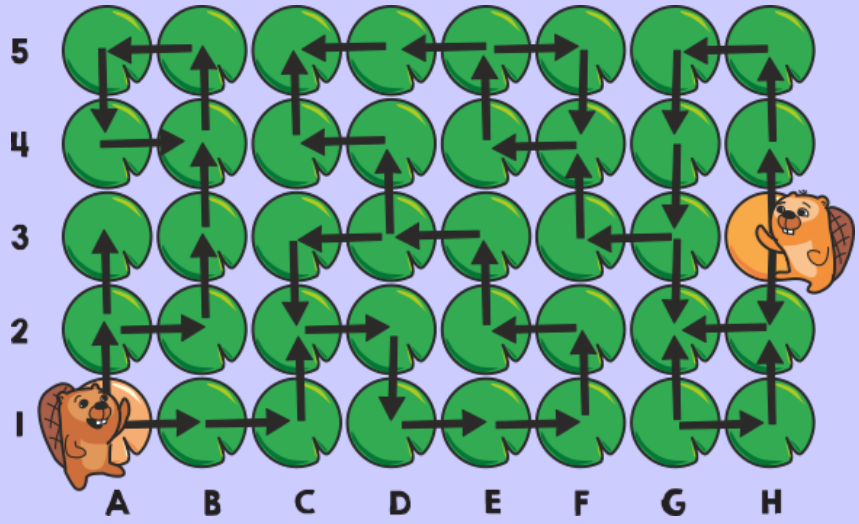
Slovakia

Part C

Do They Meet?

Story

On Lake Castor, lilypads are arranged in a grid, where rows are numbered from 1 to 5, and columns are labelled from A to H. Beaver Bob starts on pad A1 (in the bottom-left corner), and Beaver Nora starts on pad H3.



The beavers can move from one lilypad to another lilypad only if they are following an arrow. The beavers do not necessarily move at the same speed.

Question

Which of the following statements is true?

(A) The beavers will never meet.

(B) The beavers could meet on pad C2.

(C) The beavers could meet on pad F4.

(D) The beavers could meet on pad C5.

Answer

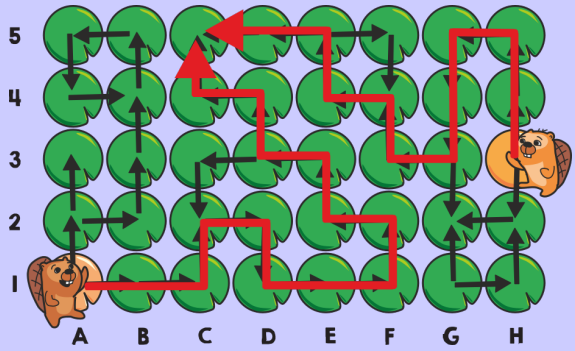
(D) The beavers could meet on pad C5.

Explanation of Answer

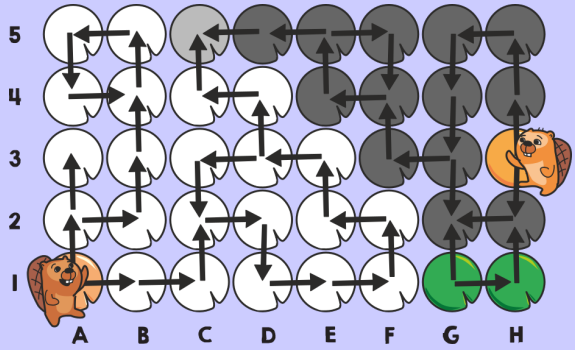
At his starting position, Bob has two options. If he goes "up", then he will either run into the dead end at A3 or get stuck in the loop that begins at B4. If he goes "right", he will follow the arrows to D3. At D3 he can either go to C3 which goes into a loop that will take him back to D3, or go to D4, which leads to a dead end at C5.

Nora also has two options at the start. If she goes "down", she will run into the dead end at G2. If she goes "up", she will reach G3. From there she can either run into the G2 dead end again, or go "left" and reach E5. There she can either go into a loop that will take her back to E5, or reach another dead end at C5.

Since Bob can reach C5 as well, the beavers could meet at C5 using the routes shown below.



But this does not yet fully guarantee that they cannot meet at F4 or C2 either. The next picture shows the set of pads that Bob (white) and Nora (dark gray) can reach by following the arrows in any possible way. We can see that C5 is the only pad common to both sets.



Connections to Computer Science

One way to find a solution to this problem is to apply a *depth first search (DFS)* algorithm, which is also known as a *backtracking* algorithm. This technique tries to reach the destination by "moving forward" until either the solution is found, or there are no more possible "forward" movements, at which point we "backtrack" to an earlier position and try another path forward.

This backtracking concept is a technique used in many applications: solving puzzles like sudoku, determining possible moves in games like chess, or combinatorial optimization problems like how to ship packages to a variety of locations as cost-effectively as possible.

Country of Original Author

Lithuania

Paintings

Story

Paintings are brought to a warehouse for inspection before they are delivered to museums. The paintings are stacked on top of each other. When a painting arrives at the warehouse, it is put on top of the stack. When a delivery person departs with a painting, they take the painting from the top of the stack.

Records are kept of all paintings arriving at the warehouse and departing from the warehouse:

Arrivals

Time	**Painting**
11:40	Beavers on the Grass
12:15	Happy Beaver
12:55	Sun and Moon
13:30	Enchanted Forest
14:18	Oak and Birch

Departures

Time	**Delivery Person**
12:25	Pia
13:35	Raz
14:35	Stu
14:40	Quy
15:20	Raz

Question

Which delivery person took "Sun and Moon" to a museum?

(A) Pia

(B) Quy

(C) Raz

(D) Stu

Answer

(B) Quy

Explanation of Answer

There are two important types of events: a painting is put on the stack and a delivery person takes a painting from the top of the stack. From the two given tables, we build a single new table that displays both types of events together and the state of the stack after each event. We add rows to the table in chronological order stopping when we learn which delivery person took "Sun and Moon" to a museum.

Time	**Event**	**Paintings on the stack**
11:40	Arrival of Beavers on the Grass	Beavers on the Grass
12:15	Arrival of Happy Beaver	Happy Beaver Beavers on the Grass
12:25	**Pia** takes Happy Beaver	Beavers on the Grass
12:55	Arrival of Sun and Moon	Sun and Moon Beavers on the Grass
13:30	Arrival of Enchanted Forest	Enchanted Forest Sun and Moon Beavers on the Grass
13:35	**Raz** takes Enchanted Forest	Sun and Moon Beavers on the Grass
14:18	Arrival of Oak and Birch	Oak and Birch Sun and Moon Beavers on the Grass
14:35	**Stu** takes Oak and Birch	Sun and Moon Beavers on the Grass
14:40	**Quy** takes Sun and Moon	Beavers on the Grass

Connections to Computer Science

There are three computer science concepts used in this task.

The first concept is the use of a *stack*. In this task, the paintings are placed on a stack, but there is also the computer science *data structure* called a stack, which relies on the model of *Last In, First Out (LIFO)*.

The second concept is the concept of *merging sorted data*. To reach the solution, the two sorted lists of events which are sorted by timestamps were merged into one sorted list. This process is one key component of the *merge sort* algorithm, which relies on the *divide and conquer* problem solving technique.

The third concept is the concept of *tracing the execution of a program*. We can consider the arrivals and departures of paintings as a *sequence of instructions* in a program. To find the delivery person who took the "Sun and Moon" painting, we can keep track of the *state* of the program after each instruction (arrival or departure) is executed, and once the desired state is reached (when the "Sun and Moon" painting is picked up), we can stop tracing.

Country of Original Author

Slovenia

Shapes

Story

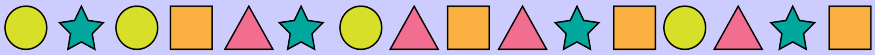
Here is a line of shapes.

The line has a run of stars of length 2. A *run* is an unbroken chain of identical shapes.

Ali likes to create long runs by changing shapes. For example, if Ali changes the middle square to a star in the line above, then he can create a longer run of length 4.

Question

Suppose Ali chooses and changes exactly 3 of the 16 shapes in the following line:



What is the length of the longest possible run that Ali can create?

(A) 4

(B) 5

(C) 6

(D) 7

Answer

(C) 6

Explanation of Answer

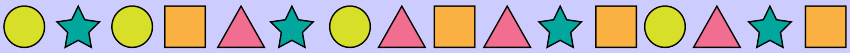
To show that Option C is correct, we need to prove two things:

(1) that a run of length 6 is possible, and

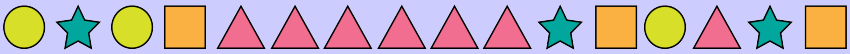
(2) that a run with length greater than 6 is not possible.

A run of length 6 can be made by changing the second star, third circle, and second square to triangles, as shown.

Before:

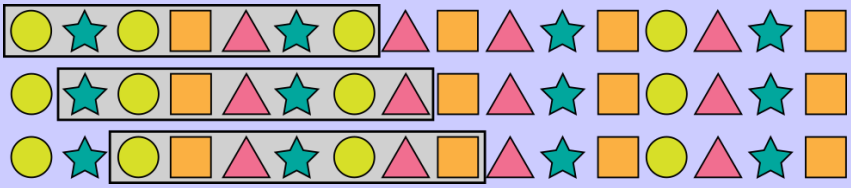


After:



To prove that a run with length greater than 6 is not possible, consider what it would mean if there was a longer run. It would mean there is a run of length 7. Remember that Ali changes only 3 shapes. Therefore, any run of length 7 must come from an unbroken chain of length 7 with 4 identical shapes in it.

There are ten unbroken chains of 7 shapes in the original sequence, the first few of which are shown below.



As we can see, none of these unbroken chains of length 7 have 4 identical shapes in them. Convince yourself that this is also true for the remaining unbroken chains of length 7. This means that it is not possible to have a run of length 7.

We have shown there are 6 shapes in the longest possible run.

Connections to Computer Science

This task is related to finding a *longest substring* that matches some given criteria. A *substring* is a sequence of adjacent characters, such as "ubs" within "substring".

One of the most studied applications of the longest substring in computer science is the *longest common substring* between two strings. Finding the longest common substring is used in *plagiarism detection* to determine if a document contains a significant amount of copied material. Another application of longest common substring is in *data deduplication*, where redundant copies of data (such as duplicate files on a disk drive) are found and removed.

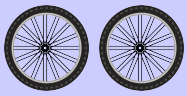
Country of Original Author

Ukraine

Upcycling

Story

Doreen uses old things as supplies to make new items; this is called *upcycling*. Doreen upcycles her supplies into wheels, bicycles, and tricycles, then sells her new items at the market. The supplies needed to make each new item are shown in the table.

Supplies Needed			New Item
 Tire	+	 Iron Bar	 Wheel
 2 Wheels	+	 Iron Bar	 Bicycle
 Bicycle	+	 Wheel	 Tricycle

Question

Doreen has 9 tires and 11 iron bars. What is the maximum number of tricycles she can make?

(A) 1

(B) 2

(C) 3

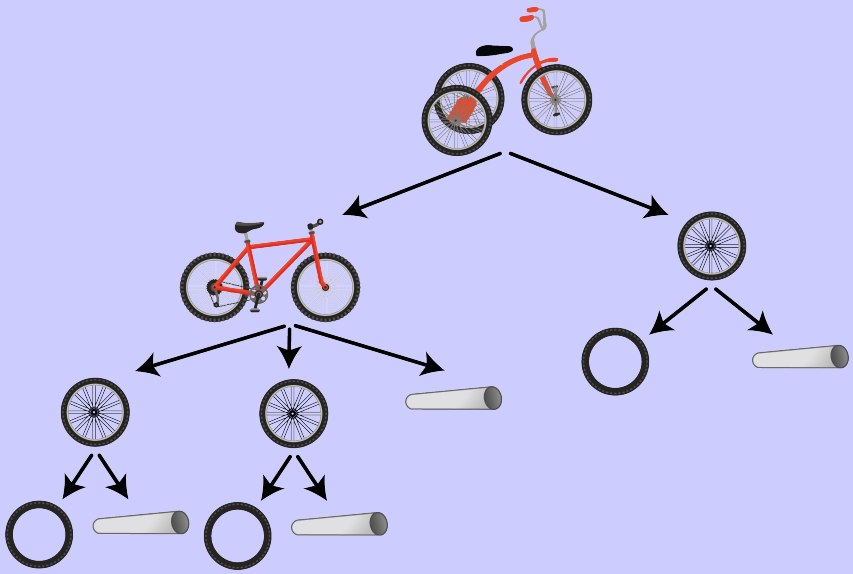
(D) 4

Answer

(B) 2

Explanation of Answer

First we will work backwards to determine how many tires and iron bars are needed to make one tricycle. The tree diagram below illustrates this process.



As we can see, one tricycle requires 3 tires and 4 iron bars. This means two tricycles require 6 tires and 8 iron bars, and three tricycles require 9 tires and 12 iron bars.

Doreen has 9 tires and 11 iron bars, so she can make two tricycles but doesn't have enough iron bars to make three tricycles.

Connections to Computer Science

Efficient use of resources is a common problem in society, and computer scientists can write *optimization algorithms* that make the most efficient choices as possible.

One strategy for an optimization algorithm is to use a *greedy strategy*. This task can be solved using a greedy strategy. Doreen builds as many wheels as possible, then as many bicycles as possible, and finally, as many tricycles as possible.

More generally, a greedy strategy involves making the best choice at each stage in the hopes of finding the best solution. In this way, it makes one greedy choice after another, reducing the given problem into a smaller one.

However, not every problem can be solved optimally using a greedy strategy. For example, *knapsack problems* usually do not have optimal solutions that can be found using a greedy strategy. Nevertheless, the greedy strategy is still useful because it is easy to describe and implement and often gives a good approximation to the optimal solution.

Country of Original Author

United Kingdom